Aims

This exercise aims to get you to practice more on Spark Programming.

Background

The transformation and action functions examples are available at:

http://homepage.cs.latrobe.edu.au/zhe/ZhenHeSparkRDDAPIExamples.html

Scala string interpolation methods (used to format your output) can be found at:

https://docs.scala-lang.org/overviews/core/string-interpolation.html

A tutorial of Scala is available at:

http://docs.scala-lang.org/tutorials/?_ga=1.99469143.850382266.1473265612

Spark Core Programming

Question 1. Download the input text file pg100.txt from: https://webcms3.cse.unsw.edu.au/COMP9313/18s2/resources/20196. Compute the average length of words starting with each letter. This means that for every letter, you need to compute: the total length of all words that start with that letter divided by the total number of words that start with that letter.

- Ignore the letter case, i.e., consider all words as lower case.
- Ignore terms starting with non-alphabetical characters, i.e., only consider terms starting with "a" to "z".
- The length of a term X can be obtained by X.length.
- Use the following split function to split the documents into terms:

```
split("[\\s*$&#/\"'\\,.:;?!\\[\\](){}<>~\\-_]+")
```

Your Spark program should generate a list of key-value pairs. Keys and values are separated by ",", and the values are of double precision, ranked in alphabetical order. You can see the result at: https://webcms3.cse.unsw.edu.au/COMP9313/18s2/resources/17756.

Name your scala file as "Problem1.scala", the object as "Problem1", and put it in a package "comp9313.lab7". Put the input file in HDFS folder "/user/comp9313/input", and store your output in HDFS folder "user/comp9313/output". The input and output paths are obtained from the arguments.

Question 2. Download the sample input file "Votes.csv" from: https://webcms3.cse.unsw.edu.au/COMP9313/18s2/resources/17758, and put it in HDFS folder "/user/comp9313/input". In this file, the fields are separated by ',' and the lines are separated by '\n'. The data format of "Votes.csv" is as below:

```
      - Id
      - PostId
      - VoteTypeId
         - ` 1`: AcceptedByOriginator
         - ` 2`: UpMod
         - ` 3`: DownMod
         - ` 4`: Offensive
         - ` 5`: Favorite - if VoteTypeId = 5 UserId will be populated
         - ` 6`: Close
         - ` 7`: Reopen
         - ` 8`: BountyStart
         - ` 9`: BountyClose
         - `10`: Deletion
         - `11`: Undeletion
         - `12`: Spam
         - `13`: InformModerator
         - `14`:
         - `15`:
         - `16`:
      - UserId (only for VoteTypeId 5)
      - CreationDate
```

(i). Find the top-5 VoteTypeIds that have the most distinct posts. You need to output the VoteTypeId and the number of posts. The results are ranked in descending order according to the number of posts, and each line is in format of: VoteTypeId\tNumber of posts.

(ii). Find all posts that are favoured by more than 10 users. You need to output both PostId and the list of UserIds, and each line is in format of:

PostId#UserId1,UserId2,UserId3,…,UserIdn

The lines are sorted according to the NUMERIC values of the PostIds in ascending order. Within each line, the UserIds are sorted according to their NUMERIC values in ascending order.

(Hint: the mkString function is useful to format your output)

You can download the code template at: https://webcms3.cse.unsw.edu.au/COMP9313/18s2/resources/17755.

You can see the result at: https://webcms3.cse.unsw.edu.au/COMP9313/18s2/resources/17757.